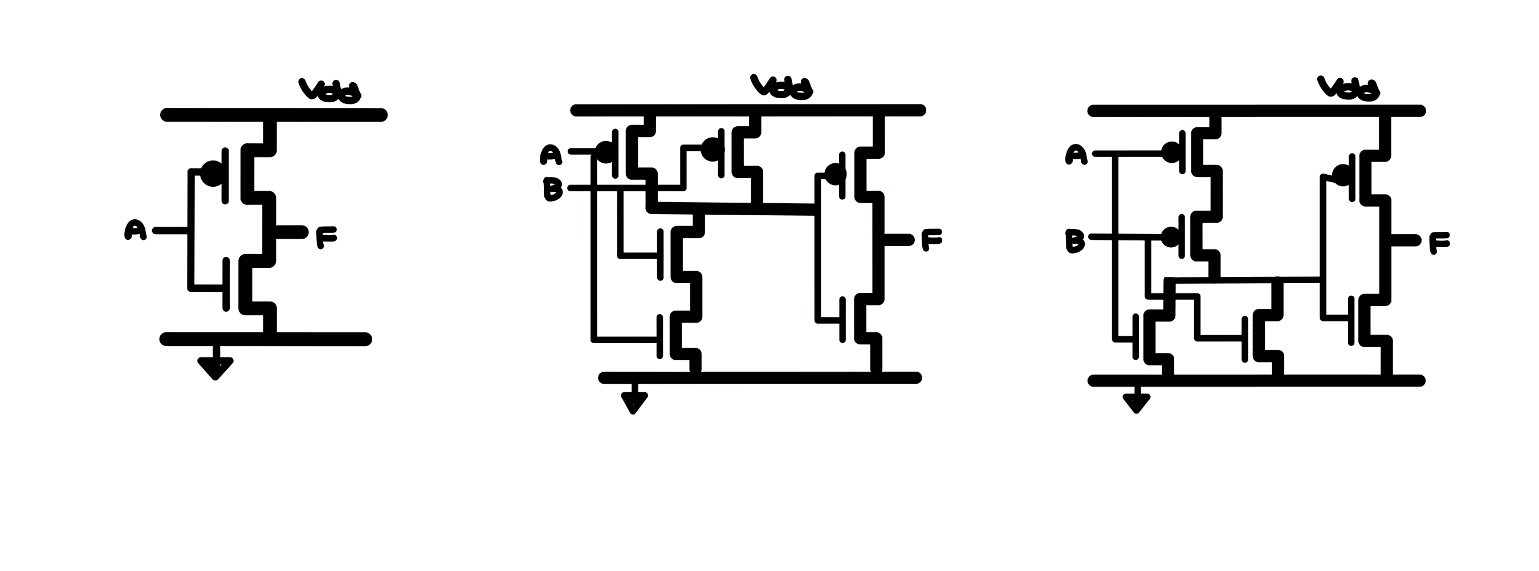
3주차 예비보고서

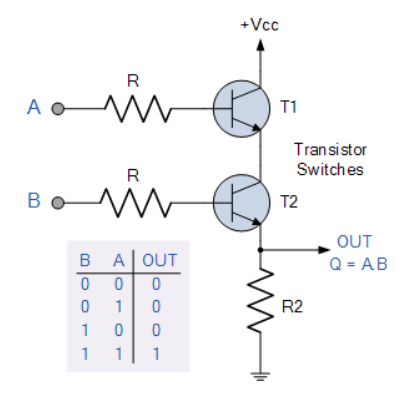
전공: 물리학과 학년: 3학년 학번: 20191286 이름: 김나현

1. 논리 게이트 AND/OR/NOT의 구조를 Transistor-level로 그리시오.

<Figure 1> transistor-level의 not, and, or 게이트의 구조

논리회로에서 사용되는 주로 사용되는 두 개의 상태는 1과 0인데, 1상태는 0상태보다 전압이 높다는 것을 명확하게 표기하기 위해 high라고 불리기도 하고, 반대로 0상태는 1상태보다 전압이 낮다는 것을 표기하기 위해 low라고 불리기도 합니다.

<Figure 1>보다 자세한 회로 구조를 확인하기 위해 AND 게이트를 예시로 들어보면, AND 게이트와 같은 역할을 수행하기 위해서는 트랜지스터, 저항, 전원 등을 이용하여 다음과 같이 회로를 구성해야 합니다.



<Figure 2> 저항과 트랜지스터를 이용한 AND 게이트

이때, T1과 T2에 인가되는 전압에 따라 입력이 0인지, 1인지 판단되므로 A와 B에서 인가하는 전압의 크기가 입력 값을 결정하고 실제로 회로에서는 0과 1을 나타내는 것이 전압의 크기라는 것을 확인할 수 있습니다.

1. AND/OR/NOT logic의 특성에 대해 조사하시오.

AND 게이트는 두 개 이상의 입력과 단일한 출력을 갖는 논리회로입니다. AND 게이트는 모든 입력이 1일 때만 1을 출력하고 입력 중에 0이 하나라도 있으면 0을 출력하게 됩니다. 일반적으로 두 개, 네 개, 여덟 개의 입력에 대해 하나의 출력을 갖는 AND 게이트가 널리 이용됩니다. 이처럼 AND 게이트에 입력이 무조건 두 개는 아니지만 AND 게이트의 연산을 알아보기 위해 다음 표와 같이 a와 b의 두 개의 입력을 갖는 AND 게이트의 진리표를 확인해볼 수 있습니다.

<Table 1> 두 개의 입력을 갖는 AND 게이트의 진리표

|  |  |  |
| --- | --- | --- |
| 입력 | | 출력 |
| A | B | Y |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

이때, 진리표에서 n개의 입력 변수에 대한 가능한 조합의 개수 N은 으로 나타낼 수 있고 두 개의 입력에 대한 가능한 조합의 개수는 위의 표에 나와 있듯이 4라는 사실을 직접 확인할 수 있습니다. 따라서 두 개의 입력이 아닌, 4개, 8개의 입력에 대해서는 각각 16, 256개의 출력 결과를 확인할 수 있습니다. AND 게이트는 논리곱 게이트라고도 불리는데, 이는 AND 게이트가 곱셈 규칙을 이용하기 때문입니다. 그러므로 AND 게이트의 출력은 방정식 Y=A·B 또는 Y=AB라고 표현할 수 있는데 0과 1에 대한 곱셈처럼 입력 A, B 중 하나만이라도 0이면 Y는 0이 되고 두 입력 모두 1인 경우에만 출력 Y가 1이 됩니다.

OR 게이트는 입력 중 하나라도 1이면 출력이 1이 되고, 모든 입력이 0일 경우에만 출력이 0이 됩니다. OR 게이트 역시 두 개의 입력뿐만 아니라 3개, 4개, 8개 등 다양한 개수의 입력이 존재할 수 있지만 두 개의 입력을 갖는 상황에서의 진리표를 확인해보면 다음과 같습니다.

<Table 2> 두 개의 입력을 갖는 OR 게이트의 진리표

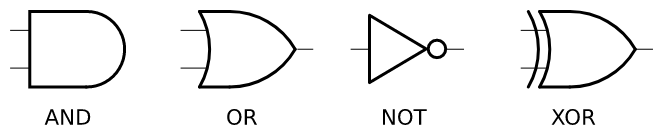
|  |  |  |
| --- | --- | --- |
| 입력 | | 출력 |
| A | B | Y |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

OR 게이트는 다른 말로 논리합 게이트라고 불리는데, 이 역시 OR 게이트가 기본적인 덧셈 연산과 관련이 있기 때문입니다. OR 게이트의 출력은 Y=A+B라고 표현할 수 있고, 실제 덧셈에서는 두 입력이 모두 1인 경우, 출력 Y가 2가 되지만 여기서 사용하는 ‘+’ 표현은 일반적인 수학과는 다른 것이므로 두 입력이 모두 1인 경우에 출력 Y는 1이 됩니다.

AND 게이트와 OR 게이트와 달리, 어떠한 경우에서도 단 한 개의 입력을 갖는 NOT 게이트가 있습니다. NOT 게이트는 역함수(inversion function) 또는 보함수(complete-mentation function)의 기능을 수행하며 입력 값을 뒤집는 역할을 하므로 부정회로 또는 인버터(inverter)라고 불립니다. NOT 게이트의 목적은 출력 상태를 입력 상태와 반대로 만드는 것입니다. NOT 게이트의 입력을 A라고 했을 때, 출력 Y는 A 문자 위에 바를 그어 로 표현합니다.

<Table 3> NOT 게이트의 진리표

|  |  |
| --- | --- |
| 입력 | 출력 |
| A | Y |
| 0 | 1 |
| 1 | 0 |



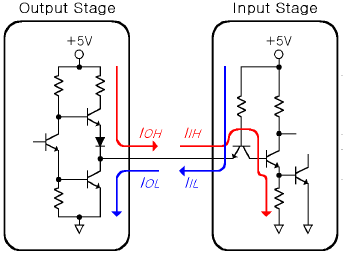
<Figure 3> AND, OR, NOT 게이트의 논리 기호

위의 <Figure 2>는 왼쪽부터 AND, OR, NOT 게이트의 논리 기호를 보여줍니다. AND, OR, NOT의 출력을 나타내는 방법으로는 언급한 방법 외에도 각각

이 있습니다.

1. Fan-out에 대해 조사하시오.

Fan-out이란 논리회로에서 한 게이트의 출력이 다른 게이트의 입력으로 사용된 수를 의미하며 이는 해당 게이트가 감당할 수 있는 출력 전력과 연관이 있습니다. 일반적으로 디지털 회로에 사용되는 표준 논리소자들(TTL, CMOS 등)은 1개의 출력 신호에 접속할 수 있는 입력 신호의 수가 제한적입니다. Fan-out이 크다는 말은 하나의 출력이 많은 논리 게이트의 입력으로 사용될 수 있다는 의미이고, fan-out이 너무 크면 출력 신호가 약해 다른 논리 게이트에 입력되지 않을 수도 있고 출력 전류가 너무 커서 출력단의 회로에 무리가 갈 수도 있습니다. 따라서 중간에 버퍼나 NOT 게이트 두 개를 연결하여 이러한 문제를 해결하기도 합니다.

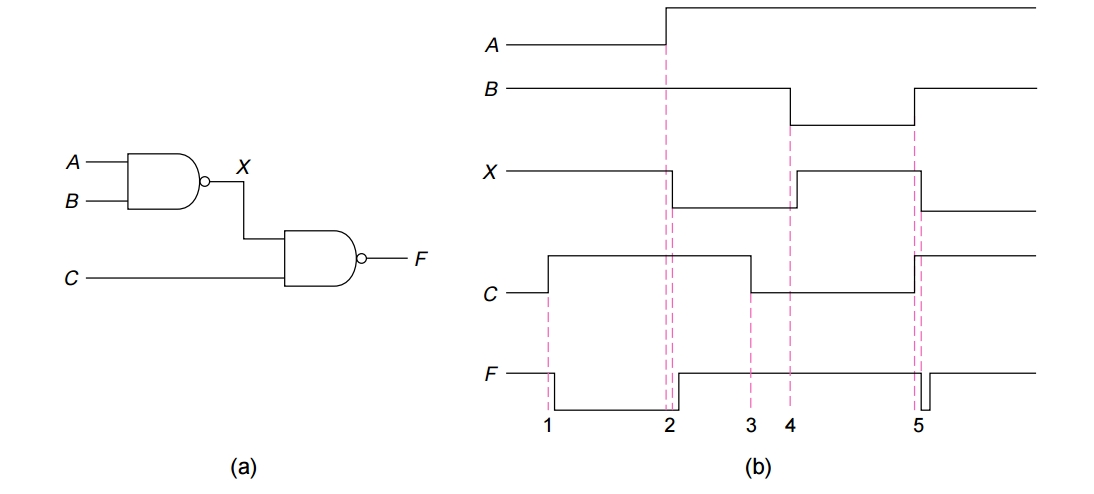


<Figure 4>TTL fan-out 회로

Fan-out은 출력단의 output current와 입력단의 input current의 비에 의해서 다음과 같이 결정이 됩니다.

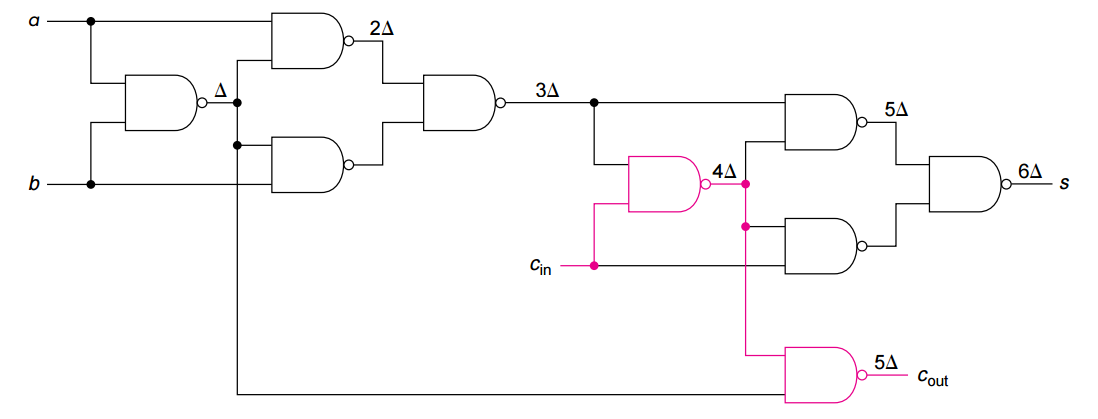
따라서 만약 <Figure 4>와 같이 TTL 회로에서 =0.4mA, =0.04mA이면 fan-out은 10이기 때문에 표준 TTL 소자에는 표준 TTL 소자가 10개까지 부하 접속이 가능하다는 뜻입니다.

1. 전파지연에 대해 조사하시오.



<Figure 5> gate delay를 확인하기 하기 위한 논리 회로(a)와 timing diagram(b)

<Figure 5>의 (a) 회로는 NAND 게이트를 이용한 회로로, NAND 게이트에 대해 간략히 말하면 이름과 게이트의 기호를 보면 알 수 있듯이 동일한 입력에 대해 AND 게이트와 정반대의 출력을 보이는 게이트라고 생각할 수 있습니다. 따라서 AND 게이트와 반대로, 모든 입력이 1일 때만 0을 출력하고 입력 중 하나라도 0이 있으면 1을 출력하는 것입니다. 따라서 (b) timing diagram을 보면 입력 A와 B가 0과 1일 때는 X가 1이다가, 입력 A가 1이 되면서 X는 0으로 바뀝니다. 이때, A가 바뀌자마자 출력 값 X가 바뀌는 것이 아니라 X가 바뀌는 데에 약간의 delay가 있다는 것을 볼 수 있습니다. 마찬가지로 X와 C가 1과 0일 때는 F가 1이다가, 입력 C가 1이 되면서 F는 0이 되는데 그때에도 바로 바뀌는 것이 아니라 약간의 delay가 나타납니다.



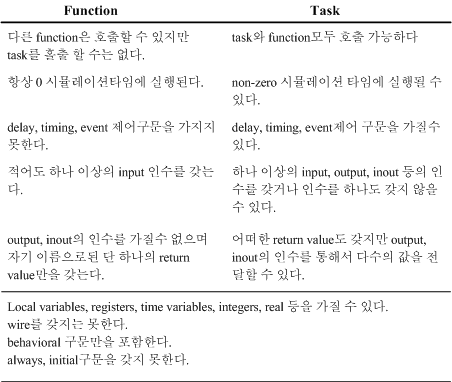
<Figure 6> 1-bit adder

그렇다면 NAND 게이트를 이용하여 위의 <Figure 6>와 같이 만든 회로에서는 각각의 게이트에서의 delay가 모여 더욱이 입력 값 a와 b의 변화가 출력 s에 바로 적용되어 나타나지 않게 될 것입니다. 이러한 것을 전파지연(propagation delay)이라고 하며, 신호 값의 변화가 입력에서 출력까지 전달되는 데에 걸리는 평균 시간을 의미합니다. 이때, 위에서도 확인하였듯이 전체 회로의 전파지연은 논리 게이트의 지연시간, 즉 한 개의 논리 게이트에서 나타나는 지연시간과 논리 게이트의 개수 등에 의해 영향을 받습니다. 입력 값에 따라 출력이 1에서 0이 되는 경우와 0에서 1이 되는 경우가 존재하므로 각각의 경우에 대해 지연시간을 , 이라고 하면 회로 전체의 지연시간 는 두 지연시간의 평균값인

으로 구할 수 있습니다.

1. Verilog의 task 및 function에 대해 조사하시오.

Verilog의 task는 정의된 모듈 안에서 언제든지 호출할 수 있는 서브루틴 keyword입니다. Task를 사용하는 방법은 모듈을 선언하는 것과 비슷하게 task와 endtask라는 keyword를 이용합니다. Task는 # delay, posedge clk, negedge clk과 같이 시간 지연과 관련된 기능을 수행할 수 있고, 입력과 출력의 개수 제한이 없으며 변수가 task 내부에서 선언될 경우, 해당 변수는 task의 지역 변수이 task 밖에서 사용할 수 없다는 특징이 있습니다. 또한 task는 지역 변수가 선언되지 않으면 전역 변수를 사용할 수 있으며 내부에 task나 function을 포함할 수도 있습니다. 뿐만 아니라 Task는 명령문으로 호출이 가능하며 표현식 내부에서는 사용이 불가능합니다.

Verilog의 function은 앞서 설명한 task와 비슷하지만 약간의 차이가 존재합니다. 모듈과 task처럼 function 또한 function과 endfunction이라는 keyword를 사용하여 선언합니다. Function은 입력과 출력의 개수 제한이 없는 task와 달리, 둘 이상의 출력을 구동할 수 없고 시간 지연 관련 기능을 수행할 수 없습니다. 또한, 표현식 내부에서 사용할 수 없는 task와 달리, function은 표현식 내부에서도 사용이 가능합니다. 내부에서 task와 function 모두를 호출할 수 있는 task와 달리, function은 내부에서 task를 호출할 수 없고, 오직 다른function만을 호출할 수 있습니다. 반대로, Function과 Task은 공통점도 존재한데 그것은 function 또한 내부에서 변수(지역 변수)를 선언할 수 있고, 지역 변수를 사용하지 않을 때에 전역 변수를 사용하고 수정할 수 있다는 점입니다.

<Figure 7> function과 task의 공통점과 차이점